

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**



JPA2002-196905

## PATENT ABSTRACTS OF JAPAN

(11) Publication number: **2002196905 A**

(43) Date of publication of application: **12.07.02**

(51) Int. Cl

**G06F 3/12**  
**B41J 29/38**

(21) Application number: 2000391377

(71) Applicant: **CANON INC**

(22) Date of filing: 22.12.00

(72) Inventor: **KAWAMOTO KOICHI**

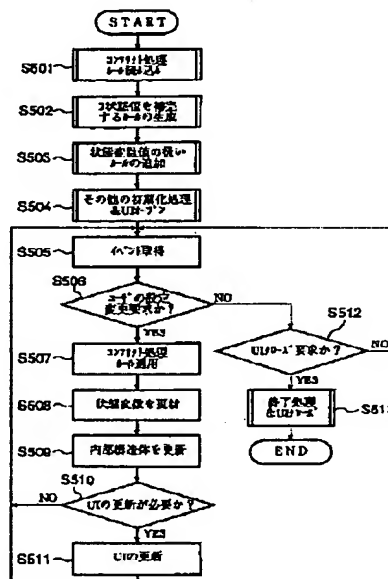
**(54) INFORMATION PROCESSING UNIT AND METHOD, IMAGE FORMING DEVICE, AND STORAGE MEDIUM**

(57) Abstract:

**PROBLEM TO BE SOLVED:** To attain complete elimination of conflict conditions by developing a rule of complementing a basic conflict management rule.

**SOLUTION:** Basic processing for mutually matching setting conditions is performed and, to match the setting conditions, the rule of complementing the basic processing is developed (S501, S502). The setting conditions are matched in accordance with the basic processing and the complementing rule to determine control parameters based on the conditions (S507, S508).

COPYRIGHT: (C)2002,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-196905

(P2002-196905A)

(43) 公開日 平成14年7月12日 (2002.7.12)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード <sup>*</sup> (参考)
G 0 6 F 3/12		G 0 6 F 3/12	C 2 C 0 6 1
B 4 1 J 29/38		B 4 1 J 29/38	Z 5 B 0 2 1

審査請求 未請求 請求項の数14 O L (全 10 頁)

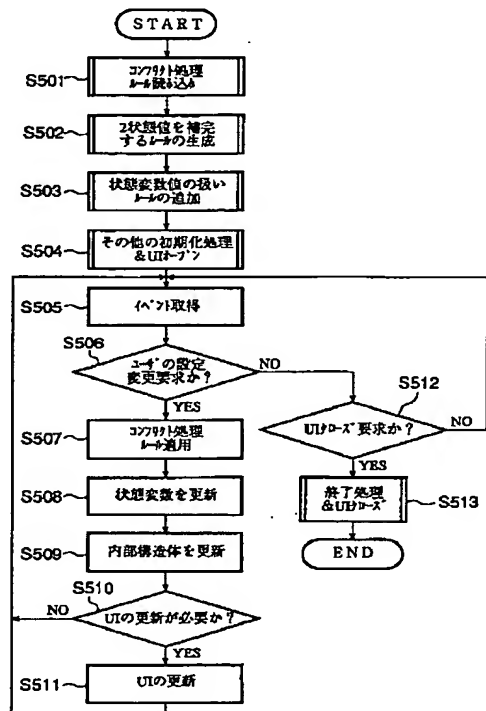
(21) 出願番号	特願2000-391377(P2000-391377)	(71) 出願人	000001007 キヤノン株式会社 東京都大田区下丸子3丁目30番2号
(22) 出願日	平成12年12月22日 (2000. 12. 22)	(72) 発明者	川本 浩一 東京都大田区下丸子3丁目30番2号 キヤ ノン株式会社内
		(74) 代理人	100076428 弁理士 大塚 康徳 (外2名) Fターム(参考) 2C061 AP01 AP03 HV01 5B021 AA01 BB01 CC05

(54) 【発明の名称】 情報処理装置及び情報処理方法、画像形成装置、記憶媒体

(57) 【要約】

【課題】 基本コンフリクト処理ルールを補完するルールを生成して、漏れなくコンフリクト状態を解消する。

【解決手段】 設定条件を相互に整合させるための基本処理を実行し、設定条件を整合させるために、その基本処理を補完するための補完処理ルールを生成し (S501、S502)、その基本処理及び補完処理ルールに従い、設定条件を整合させて、その条件に基づく制御パラメータを決定する (S507、S508)。



## 【特許請求の範囲】

【請求項1】 設定条件を相互に整合させるための基本処理をする手段と、  
前記設定条件を整合させるために、前記基本処理を補完するための補完処理ルールを生成する生成手段と、  
前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御手段と、  
を備えることを特徴とする情報処理装置。

【請求項2】 前記制御手段は、設定条件を入力するための入力手段から入力された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする請求項1に記載の情報処理装置。

【請求項3】 設定条件を可視化するためのインタフェース手段と、  
前記制御手段により決定された条件を前記インタフェース手段に表示する表示制御手段と、  
を備えることを特徴とする請求項1または2に記載の情報処理装置。

【請求項4】 前記表示制御手段は、前記制御手段による前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする請求項3に記載の情報処理装置。

【請求項5】 請求項1乃至4のいずれかに記載の情報処理装置と、  
前記情報処理装置に入力された画像を形成するための制御パラメータを決定し、該決定された制御パラメータに基づき画像情報を形成するための画像形成手段と、  
を備えることを特徴とする画像形成装置。

【請求項6】 前記画像形成装置にはプリンタ、ファクシミリが含まれることを特徴とする請求項5に記載の画像形成装置。

【請求項7】 設定条件を相互に整合させるための基本処理を実行する工程と、  
前記設定条件を整合させるために、前記基本処理を補完するための補完処理ルールを生成する生成工程と、  
前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御工程と、  
を備えることを特徴とする情報処理方法。

【請求項8】 前記制御工程は、設定条件を入力処理するための入力工程から入力された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする請求項7に記載の情報処理方法。

【請求項9】 設定条件を可視化させるためのインタフェース工程と、

前記制御工程により決定された条件を前記インタフェース工程で表示処理させる表示制御工程と、  
を備えることを特徴とする請求項7または8に記載の情報処理方法。

【請求項10】 前記表示制御工程は、前記制御工程による前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする請求項9に記載の情報処理方法。

【請求項11】 情報処理方法をコンピュータで実行するためのプログラムモジュールを格納したコンピュータ可読の記憶媒体であって、該プログラムモジュールが、  
設定条件を相互に整合させるための基本処理を実行するモジュールと、  
前記設定条件を整合させるために、前記基本処理を補完するための補完処理ルールを生成する生成モジュールと、  
前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御モジュールと、  
を備えることを特徴とする記憶媒体。

【請求項12】 前記制御モジュールは、設定条件を入力処理するための入力モジュールで入力処理された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする請求項11に記載の記憶媒体。

【請求項13】 設定条件を可視化処理するためのインタフェースモジュールと、  
前記制御モジュールにより決定された条件を前記インタフェースモジュールで表示処理させる表示制御モジュールと、  
を備えることを特徴とする請求項11または12に記載の記憶媒体。

【請求項14】 前記表示制御モジュールは、前記制御モジュールによる前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする請求項13に記載の記憶媒体。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】本発明は、情報処理装置の処理条件の設定において、複数の設定項目間で条件の不整合が生じた場合に、その不整合を解消する装置及び情報処理方法及びその方法をコンピュータで実行するためのプログラムモジュールを格納した記憶媒体、その情報処理装置を組み込んだ画像形成装置に関する。

## 【0002】

【従来の技術】従来、コンピュータ上で動作するプログラムはユーザインタフェースを介してユーザから複数の動作内容を定義するための設定値の入力を受け付けると、それらの設定値に基づいて処理を実行する。

【0003】ところが、複数の設定条件において排他的な関係や、設定条件相互間で依存する関係等が存在する場合はユーザからの設定値の入力を受け付ける度に、入力された設定条件、設定値との関係の評価し、それらの設定条件等に不整合がないかどうかの判別を行い、不整合があった場合にはその解消を実施する処理を行っていた。その処理の実現にあたっては、不整合の検知および不整合の解消処理を設定値間の関係に依存した条件の下で専用の処理プログラムを用いるのが一般的である。

【0004】あるいは、不整合の処理が必要となる複数の設定値の条件を一覧としてまとめてファイル等に保存しておき、このファイルを不整合を解消する汎用的な処理プログラムを利用して、このプログラムに読み込ませて、設定条件の不整合を解消させる場合もある。

【0005】

【発明が解決しようとする課題】しかしながら、上記の不整合（以下、「コンフリクト」）をプログラムにより解消するための条件の記述は、プログラム開発者などが全ての条件を網羅的に行うため、設定値間の依存関係が複雑な場合には、全ての条件を網羅することができず、漏れ、すなわち、不整合の状態が完全に解消されない場合が生じることがあった。

【0006】

【課題を解決するための手段】本発明は上記従来例における課題に鑑みてなされたもので、プログラム開発者などが用意するコンフリクトを解消するための条件を基に、この条件を補完する条件を自動生成し、コンフリクト条件に加えることで、漏れがなく、より確かなコンフリクトの解消処理を実現することを目的とする。

【0007】上記目的を達成するために、本発明にかかる情報処理装置、方法およびその方法をコンピュータで実行するためのプログラムモジュールを格納した記憶媒体は主として以下の構成を有することを特徴とする。

【0008】すなわち、情報処理装置は、設定条件を相互に整合させるための基本処理をする手段と、前記設定条件を整合させるために、前記基本処理を補完するための補完処理ルールを生成する生成手段と、前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御手段と、を備えることを特徴とする。

【0009】上記の情報処理装置において、前記制御手段は、設定条件を入力するための入力手段から入力された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする。

【0010】上記の情報処理装置において、設定条件を可視化するためのインタフェース手段と、前記制御手段により決定された条件を前記インタフェース手段に表示する表示制御手段と、を備えることを特徴とする。

【0011】上記の情報処理装置において、前記表示制御手段は、前記制御手段による前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする。

【0012】また、画像形成装置において、上記の情報処理装置と、前記情報処理装置に入力された画像を形成するための制御パラメータを決定し、該決定された制御パラメータに基づき画像情報を形成するための画像形成手段とを備えることを特徴とする。

10 【0013】また、情報処理方法は、設定条件を相互に整合させるための基本処理を実行する工程と、前記設定条件を整合させるために、前記基本処理を補完するための補完処理ルールを生成する生成工程と、前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御工程と、を備えることを特徴とする。

20 【0014】上記情報処理方法において、前記制御工程は、設定条件を入力処理するための入力工程から入力された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする。

【0015】上記情報処理方法において、設定条件を可視化させるためのインタフェース工程と、前記制御工程により決定された条件を前記インタフェース工程で表示処理させる表示制御工程と、を備えることを特徴とする。

30 【0016】上記の情報処理方法において、前記表示制御工程は、前記制御工程による前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする。

40 【0017】また、情報処理方法をコンピュータで実行するためのプログラムモジュールを格納したコンピュータ可読の記憶媒体であって、該プログラムモジュールが、設定条件を相互に整合させるための基本処理を実行するモジュールと、前記設定条件を整合させるために、前記基本処理を補完するための補完処理ルールを生成する生成モジュールと、前記基本処理及び補完処理ルールに従い、前記設定条件を整合させて、該条件に基づく制御パラメータを決定する制御モジュールと、を備えることを特徴とする。

【0018】上記の記憶媒体において、前記制御モジュールは、設定条件を入力処理するための入力モジュールで入力処理された条件に対して、該設定条件相互間の不整合の有無を判断し、不整合がある場合は、前記基本処理及び補完処理ルールを適用して制御パラメータを決定することを特徴とする。

50 【0019】上記の記憶媒体において、設定条件を可視化処理するためのインタフェースモジュールと、前記制御モジュールにより決定された条件を前記インタフェー

スモジュールで表示処理させる表示制御モジュールと、を備えることを特徴とする。

【0020】上記の記憶媒体において、前記表示制御モジュールは、前記制御モジュールによる前記基本処理及び補完処理ルールの適用により、前記設定条件が変更されたことを報知することを特徴とする。

【0021】

【発明の実施形態】以下、本発明を適用するのに好適である実施形態について説明を行う。

【0022】図1は本発明が適用される実施形態において、設定された条件が整合していないとき、この不整合を解消するための情報処理（コンフリクト解消処理）が実行される印刷処理システムの構成を説明するブロック図である。

【0023】なお、特に断らない限り、本発明にかかる情報処理が実行されるのであれば、単体の機器であっても、複数の機器からなるシステムであっても、LAN、WAN等のネットワークを介して接続が為され処理が行われるシステムであっても本発明を適用できることは言うまでもない。

【0024】図1において、3000はホストコンピュータで、ROM3bのプログラム用ROMあるいは外部メモリ11に記憶された文書処理プログラム等に基づいて図形、イメージ、文字、表（表計算等を含む）等が混在した文書処理を実行するCPU1を備え、システムバス4に接続される各デバイスをCPU1が総括的に制御する。

【0025】また、プログラム用ROM3bあるいは外部メモリ11には、CPU1の制御プログラムであるオペレーティングシステムプログラム（以下OS）等を記憶し、フォント用ROM3aあるいは外部メモリ11には上記の文書処理の際に使用するフォントデータ等が記憶され、データ用ROM3cあるいは外部メモリ11には文書処理等を行う際に使用する各種データが記憶されている。

【0026】2はRAMで、CPU1の主メモリ、ワークエリア等として機能する。5はキーボードコントローラ（KBC）で、キーボード9や不図示のポインティングデバイスからのキー入力を制御する。6はCRTコントローラ（CRTC）で、CRTディスプレイ（CRT）10の表示を制御する。

【0027】7はディスクコントローラ（DKC）で、ブートプログラム、各種のアプリケーション、フォントデータ、ユーザファイル、編集ファイル、プリンタ制御コマンド生成プログラム（以下プリンタドライバ）等を記憶するハードディスク（HD）及びフロッピー（登録商標）ディスク（FD）等の外部メモリ11とのアクセスを制御する。

【0028】8はプリンタコントローラ（PRTC）で、所定の双方向性インターフェース（インターフェー

ス）21を介してプリンタ1500に接続されて、プリンタ1500との通信制御処理を実行する。なお、CPU1は、例えばRAM2上に設定された表示情報RAMへのアウトラインフォントの展開（ラスタライズ）処理を実行し、CRT10上でのWYSIWYGを可能としている。

【0029】また、CPU1は、CRT10上の不図示のマウスカーソル等で指示されたコマンドに基づいて登録された種々のウインドウを開き、種々のデータ処理を実行する。ユーザは印刷を実行する際、印刷の設定に関するウインドウを開き、プリンタの設定や、印刷モードの選択を含むプリンタドライバに対する印刷処理方法の設定を行える。

【0030】プリンタ1500において、12はプリンタCPUで、プログラム用ROM13bに記憶された制御プログラム等あるいは外部メモリ14に記憶された制御プログラム等に基づいてシステムバス15に接続される印刷部（プリンタエンジン）17に出力情報としての画像信号を出力する。また、プログラムROM13bには、CPU12の制御プログラム等を記憶する。フォント用ROM13aには出力情報を生成する際に使用するフォントデータ等を記憶し、データ用ROM13cにはハードディスク等の外部メモリ14がないプリンタの場合には、ホストコンピュータ上で利用される情報等を記憶している。

【0031】CPU12は入力部18を介してホストコンピュータとの通信処理が可能となっており、プリンタ内の情報等をホストコンピュータ3000に通知可能に構成されている。19はCPU12の主メモリ、ワークエリア等として機能するRAMで、図示しない増設ポートに接続されるオプションRAMによりメモリ容量を拡張することができるように構成されている。

【0032】なお、RAM19は、出力情報展開領域、環境データ格納領域、NVRAM等に用いられる。前述したハードディスク（HD）、ICカード等の外部メモリ14は、メモリコントローラ（MC）20によりアクセスを制御される。外部メモリ14は、オプションとして接続され、フォントデータ、エミュレーションプログラム、フォームデータ等を記憶する。

【0033】また、18は前述した操作パネルで操作のためのスイッチおよびLED表示器等が配されている。また、前述した外部メモリは1個に限らず、少なくとも1個以上備え、内蔵フォントに加えてオプションフォントカード、言語系の異なるプリンタ制御言語を解釈するプログラムを格納した外部メモリを複数接続できるように構成されていてもよい。さらに、図示しないNVRAMを有し、操作パネル1501からのプリンタモード設定情報を記憶するようにしてもよい。

【0034】図2に示すのが、本実施形態におけるユーザインタフェース制御プログラムがホストコンピュータ

10

20

30

40

50

3000上のRAM2にロードされ、実行可能となった状態のメモリマップを示している。なお、本実施形態におけるユーザインタフェース制御プログラムは印刷処理関連プログラム204の一部として存在している。

【0035】図4は、本実施形態のコンフリクト処理における各モジュールで扱うデータの関連を示す図である。

【0036】図5は本実施形態におけるコンフリクトを解消するための、基本となるコンフリクトルールを補完するための補完ルールを生成するための処理（以下、「補完ルールの生成処理」という。）を説明するフローチャートである。以下、図5に示すフローチャートを中心に本実施形態を詳しく説明する。

【0037】本実施形態ではプリンタドライバがCRT10に各種設定用に表示するユーザインタフェースから設定入力された設定条件に関するコンフリクト処理を例に説明する。ユーザが図1に示したキーボード9を介して、キーボードコントローラKBC5の制御によりCRT10に表示されたユーザインタフェース画面の一例を図8に示す。プリンタドライバUI（ユーザインタフェース）を開く指示をすることでコンフリクトルールの補完処理が開始する。

【0038】ユーザがプリンタドライバUIを開く指示により、プリンタドライバUIを開くための初期化処理が行われ、OSの管理の下、RAM2に印刷処理関連プログラム204がロードされる。

【0039】補完ルールの生成処理を図3及び図6を用いて説明する。

【0040】ここで、図3は本実施形態におけるユーザインタフェース制御部の内部処理の概要を示す図である。図3において、推論エンジン302は図6（補完ルールの生成処理前のコンフリクト処理ルール）に例示してあるように表記されたコンフリクト処理ルール301をコンフリクトマネージャ303を介してRAM2中に読み込む（ステップS501）。

【0041】続いて、ステップS501で読み込んだコンフリクト処理ルールを基に、2状態値（ONとOFF）を値として有するルールについて補完ルールを生成する（ステップS502）。

【0042】例えば、以下の（1）から（3）は2状態値の設定状態を記述した例であり、同じプリンタの機能について以下の例のように左辺にONまたはOFFのどちらかだけが記述された場合を想定すると、

A(ON) <- B(ON), C(OFF)... (1)

A(ON) <- D(V1)... (2)

B(OFF) <- E(OFF)... (3)

（1）の場合は設定値BがON、設定値CがOFFに設定された場合に、設定値Aの設定がONになることを示しており、（2）は多状態設定値である設定値Dの値がV1に設定された場合も設定値Aの設定が（1）の場合

同様、ONになることを示している。また（3）は設定値EがOFFに設定された場合に、設定値Bの値がOFFになることを示している。

【0043】この場合には推論エンジン302がON/OFFを逆にするルールを以下のように自動生成する。すなわち、上記の(1), (2) に対しては（4）式のようなコンフリクト処理ルールを生成する。

【0044】つまり、AはONの状態ではなく、OFFの状態値をとる。

10 【0045】A(OFF) <- not A(ON) ... (4)

同様に(3)に対しては左辺Bの設定を変更するコンフリクト処理ルールを生成する。つまり、BはOFFの状態ではなく、ONの状態値をとる。

【0046】B(ON) <- not B(OFF) ... (5)

このように、推論エンジン302によって自動生成された(4), (5)のルールは処理効率上最適化されて以下の(4)', (5)'のように変形されるが、意味は全く同じとなる。つまり、Aの設定OFF、Bの設定ONは真であるという内容がルールとして生成されることになる。

20 【0047】

A(OFF) <- true ... (4)'

B(ON) <- true ... (5)'

上記の例のように逆の状態値に変更するように自動生成されたルールは論理的に制御対象（例えばプリンタやファクシミリ等）の設定条件を定義する状態値として100%、矛盾なく機能を遂行するための論理を整合させるものである。

【0048】続いて、図4に示すように状態変数に生成された補完ルールを反映する（ステップ503）。

30 【0049】補完ルールの生成処理において、制御対象（例えばプリンタやファクシミリ等）の全ての機能設定項目についてはコンフリクトマネージャ303の内部に、図4に示すような状態変数リスト402を持っている。この状態変数の値はプリンタドライバUIで使用する内部構造体401の対応するメンバ（状態変数）の値と連動している。全てのプリンタ機能設定項目の状態変数の初期値はその内部構造体401のメンバの値が初期値となる。

【0050】例えば以下の設定を記述した場合、

40 A(ON) <- B(ON), C(OFF)

図4の状態変数リスト402に示すようにコンフリクト処理ルール403で使用するプリンタにおける機能設定項目A, B, Cのそれぞれについて同名の状態変数が存在する。そして、プリンタにおける機能設定項目A, B, Cに対応するプリンタドライバUI内部構造体401のメンバをそれぞれint cA, int cB, int cCとする。

【0051】int cAの初期値は0であり、それに対応するAの値は0に対応する状態変数OFFとなっている。

従って推論エンジン302内のプリンタ機能設定項目Aの状態値の初期値も状態変数「OFF」が設定される。推

50

論エンジン302でコンフリクトチェックの推論が行われて以下のルール、

A(ON) <- B(ON), C(OFF)、

が成立した場合、推論エンジン302は左辺のプリンタ機能設定項目Aの状態変数値を「ON」に変更する。

【0052】コンフリクトチェックの推論が終了した後、コンフリクトマネージャは変更された状態変数の値をプリンタドライバUI内部構造体401の対応するメンバintcAに反映（マッピング）する。つまりintcAは上記ルールが成立したことによって「0」から状態変数ONに対応する「1」に変更される。

【0053】推論エンジン302は推論エンジンに組み込まれている組み関数status(a, \_X)によってプリンタ機能設定項目Aの状態変数値を推論エンジン302で使用する変数\_Xに変更する。推論エンジン302はコンフリクト処理ルールをロードした後、そのコンフリクト処理ルール中に出現する全ての状態変数について、コンフリクトを解消するための補完ルールを自動的に生成する。

【0054】

A(\_X) <- status(A, \_X)

B(\_X) <- status(B, \_X)

C(\_X) <- status(C, \_X)

これは他に適用するルールが存在しない場合にはプリンタドライバUI内部構造体401の対応するメンバの値がそのプリンタ機能設定項目の状態値となることを意味する。

【0055】プリンタの機能設定項目Aについては、

A(ON) <- B(ON), C(OFF)

なる関係が成り立と判断されればAの状態値はこれを受けてONと設定される。Bがもし上記の自動生成されたルール以外に依存するルールが存在しないならば、

B(\_X) <- status(B, \_X)

として状態変数が設定される。この自動生成ルールによりBの状態変数の値ONが\_Xにユニファイされ、それがプリンタ機能Bの状態値となる。

【0056】つまり、ユーザ定義ルールが存在しないか、または存在していてもプリンタ機能について影響が及ばない場合はプリンタドライバのUIの内部構造体401に対応するメンバに格納されている値がそのプリンタ機能の状態値ということになる。

【0057】続いて、その他の初期化処理として、プリンタドライバのUIをオープンするために必要な初期化処理を行い、図8に例示するようなプリンタドライバユーザインタフェース(UI)をオープンする（ステップS504）。プリンタドライバUIがオープンされた後は、OSより送られてくるイベントの取得とその処理を繰り返す（ステップS505）。

【0058】ステップS505にて取得したイベントが、ユーザがプリンタドライバUI上の設定項目を変更し

たイベントであるかどうかの判別を行い（ステップS506）、そうでなかった場合（S506-No）には、続いてプリンタドライバUIのクローズ要求かどうかの判別を行う（ステップS512）。クローズ要求であった場合（S512-Yes）には、終了処理を行い、プリンタドライバUIをクローズして、全ての処理を終了する（ステップS513）。一方、クローズ要求でもなかった場合（S512-No）には、再び、処理をステップS505に戻す。

10 【0059】ステップS506での判別により、ステップS505で取得したイベントがユーザの設定変更要求であった場合には、ステップS501からS503の処理により構築した補完されたコンフリクト処理ルールを適用する（ステップS507）。

【0060】ユーザの設定変更要求が図8に示すPrint Styleを「1-Sided Printing」から「Booklet Printing」に変更するものであった場合を例にとると、コンフリクト処理ルールの適用は、図6に示すコンフリクト処理ルール301の一部として読み込まれた基本となるコンフリクト処理ルールと、ステップS502にて生成された補完ルールの一部である図7(4)、(5)とステップS503にて行われた状態変数を反映する図7(6)～(9)に対して行われる。

【0061】プリンタドライバUI内部構造体401のメンバとして存在するCollate、Group、Staple、Layoutの各メンバのコンフリクト処理ルール適用前の値は、以下の通りとなる。

【0062】

Collate : OFF

30 Group : ON

Staple : OFF

Layout : 1-Sided。

【0063】ユーザの変更要求がLayoutを1-SidedからBookletに変更するものであるので、Layoutのメンバの内容は「1-Sided」から「Booklet」に変更となる。

【0064】

Collate : OFF

Group : ON

Staple : OFF

40 Layout : Booklet。

【0065】すると、プリンタドライバUI306はコンフリクトマネージャ303を呼び出し、状態変数リストにあるLayoutの状態変数が上記のように更新され、続いて推論エンジン302がコールされて、コンフリクト処理ルールの適用が始まる。

【0066】まず、図7(6)～(9)のルールが適用され、推論エンジン302内の各プリンタ機能設定項目が状態変数リストの各メンバの持つ値で初期化される。

50 【0067】続いて、図7(3)が適用され、Groupの値はONからOFFへと変更となる。



## 【0068】

Collate : OFF  
Group : OFF  
Staple : OFF  
Layout : Booklet

さらに、図7(4)のルールが適用され、CollateがOFFからONへと変更となる。

## 【0069】

Collate : ON  
Group : OFF  
Staple : OFF  
Layout : Booklet

他に、適用されるルールが存在しないので、以上で推論エンジン302でのコンフリクト処理ルールの適用が終了する。

【0070】次のステップでは、コンフリクトマネージャ303が上記の最終状態を基に状態変数リストの更新(ステップ508)とプリンタドライバUI内部構造体の更新(ステップ509)を行う。

【0071】続いて、プリンタドライバUI306がプリンタドライバUI内部構造体のメンバの値を参照して、UIの更新が必要かどうかの判別を行う(ステップ510)。

【0072】UIの更新の必要のない場合には、そのまま処理をステップ505に戻す。更新が必要な場合には、UIの更新を行い(ステップ511)、ステップ505に処理を戻す。

【0073】上記の例では、Layoutが「1-Sided Printing」から「Booklet Printing」に設定が変更されたことにより、Collateが「OFF」から「ON」へ、Groupが「ON」から「OFF」へと変化しているため、プリンタドライバUIの表示も図9に示すとおりに更新される。

【0074】以上の処理は、プリンタドライバUIがクローズされるまで、繰り返し実行される。プリンタドライバUIがクローズされると処理は全て終了し、本実施形態における印刷処理関連プログラムの処理も終了し、RAM2からはOS405の機能により消去される。なお、本実施形態においては、本印刷処理関連プログラムを記録する媒体を外部メモリとしているが、外部メモリとしては、FD、HDドライブ、CD-ROMやICメモリカード等であってもよい。更に、本印刷プログラム単独、もしくはOSその他のホストコンピュータ上で動作するプログラムと共にROM3bに記録しておき、これをメモリマップの一部となすように構成し、直接CPU1で実行することも可能である。

【0075】以上説明したように、本実施形態によれば、プログラム開発者が用意する基本となるコンフリクト処理ルールを補完するコンフリクト条件を自動生成することで、漏れのない、コンフリクト解消処理が実現できるという効果がある。

## 【0076】

【他の実施形態】上記実施形態においては、プリンタドライバUIの更新処理をプリンタドライバUI本体306で実施しているが、図10に示すように、コンフリクト処理ルールの中に、プリンタドライバUIを更新するための処理を関数として記述し、推論エンジン302がその記述を解釈した時点で、コンフリクトマネージャ303の状態変数リスト304を介して、プリンタドライバのUI更新処理を直接行うようにしても良い。

10 【0077】図10は、図7のコンフリクト処理ルールのうち、(3)にUI更新の処理を追加したものである。{disable}の記述を追加することにより、図9(プリンタドライバの表示するユーザインタフェースの例で、図8の状態、ユーザが[BookletPrinting]を選択した状態を例示する図)にあるGroupラジオボタンコントロールをdisableする処理がコンフリクト処理ルールの記述の一部として実現されるようになる。

20 【0078】さらに、図11に示すようにコンフリクト処理ルールの中に、ユーザに対する情報の表示を可能にするメッセージボックスの表示処理を加えることも可能である。

【0079】例えば、図11の(3)の{Message(MSG001)}の記述は、図12のメッセージボックスを表示するための処理を示している。MSG001は図12に表示されているメッセージテキスト「Groupの設定はCollateに調整されました。」の文字列を指し示すIDで、ID:MSG001とそのIDが示す文字列はコンフリクトマネージャ303に文字列リソースとして存在している。

30 【0080】本実施形態によれば、ユーザインタフェースの更新処理やメッセージ処理もコンフリクト処理ルールに加えることで、開発者にとって可読性の高い、メンテナンスの容易なコーディングを実現できるという効果がある。

【0081】更に、ユーザインタフェース制御部は、ユーザインタフェース更新処理とメッセージ表示をコンフリクト処理部分に設けることで、コンフリクト処理ルールが変更された場合でも容易に変更後の条件に適合した表示を実現することが可能にする。

## 【0082】

40 【発明の効果】以上説明したように、本発明によれば、基本となるコンフリクト処理ルールを補完する条件を自動生成するので、漏れがなくコンフリクト状態を解消する処理が実現できるという効果がある。

【0083】さらに他の発明によれば、ユーザインタフェースの更新処理、メッセージ処理もコンフリクト処理ルールに加えることで、開発者にとって可読性の高い、メンテナンスの容易なコーディングを実現できるという効果がある。

50 【0084】更に、ユーザインタフェース制御部は、ユーザインタフェース更新処理とメッセージ表示をコンフ

リクト処理部分に設けることで、コンフリクト処理ルールが変更された場合でも容易に変更後の条件に適合した表示を実現することが可能にする。

【図面の簡単な説明】

【図1】本発明が適用される実施形態において、コンフリクト解消処理が実行される印刷処理システムの構成を説明するブロック図である。

【図2】本実施形態におけるユーザインタフェース制御プログラムがホストコンピュータ3000上のRAM2にロードされ、実行可能となった状態のメモリマップを示している。

【図3】本実施形態におけるユーザインタフェース制御部の内部処理の概要を示す図である。

【図4】本実施形態のコンフリクト処理における各モジュールで扱うデータの関連を示す図である。

【図5】本実施形態におけるコンフリクトを解消するための、補完ルールの生成処理を説明するフローチャートである。

【図6】補完ルールの生成処理前のコンフリクト処理ルールを例示する図である。

【図7】本実施形態における補完後のコンフリクト処理

ルールを例示する図である。

【図8】本実施形態におけるプリンタドライバの表示するユーザインタフェースを例示する図である。

【図9】本実施形態におけるプリンタドライバの表示するユーザインタフェースの例で、ユーザが[Booklet Printing]を選択した状態を例示する図である。

【図10】他の実施形態におけるコンフリクト処理ルールの記述を例示した図である。

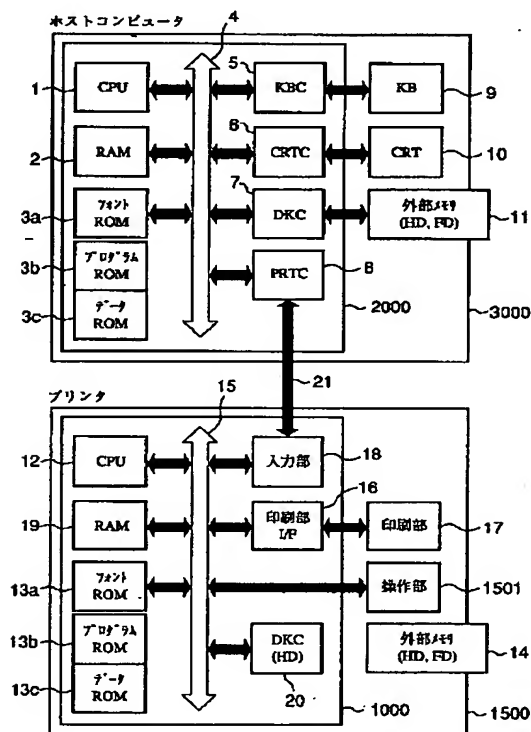
【図11】他の実施形態におけるコンフリクト処理ルールの記述を例示した図である。

【図12】他の実施形態におけるユーザに表示されるメッセージボックスを例示する図である。

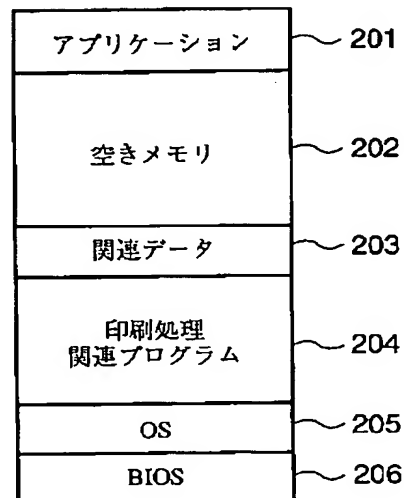
【符号の説明】

- 1 CPU
- 2 RAM
- 3 ROM
- 4 システムバス
- 12 CPU
- 13 ROM
- 20 19 RAM
- 3000 ホストコンピュータ

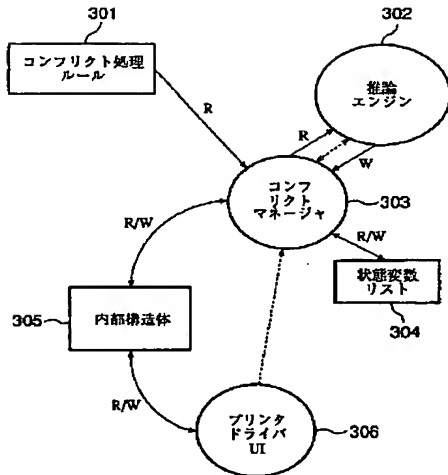
【図1】



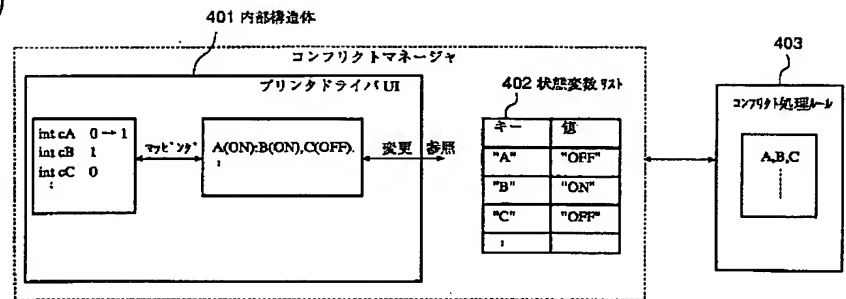
【図2】



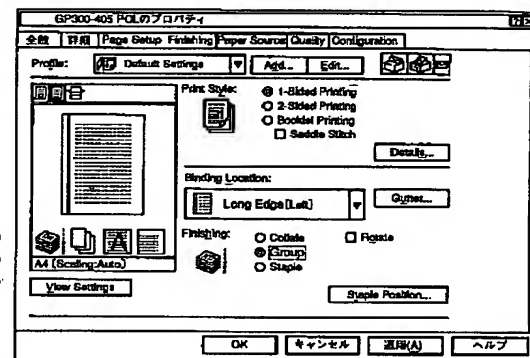
【図3】



【図4】



【図8】



【図6】

Collate(OFF) <- Group(ON).  
 Collate(OFF) <- Staple(ON).  
 Group(OFF) <- Layout(BOOKLET).

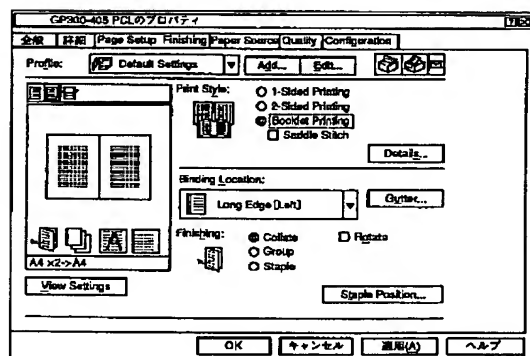
【図7】

(1) Collate(OFF) <- Group(ON).  
 (2) Collate(OFF) <- Staple(ON).  
 (3) Group(OFF) <- Layout(BOOKLET).

Collate(ON) <- true.  
 Group(ON) <- true.

Collate(\_X) <- status(Collate,\_X). (6)  
 Group(\_X) <- status(Group,\_X). (7)  
 Layout(\_X) <- status(Layout,\_X). (8)  
 Staple(\_X) <- status(Staple,\_X). (9)

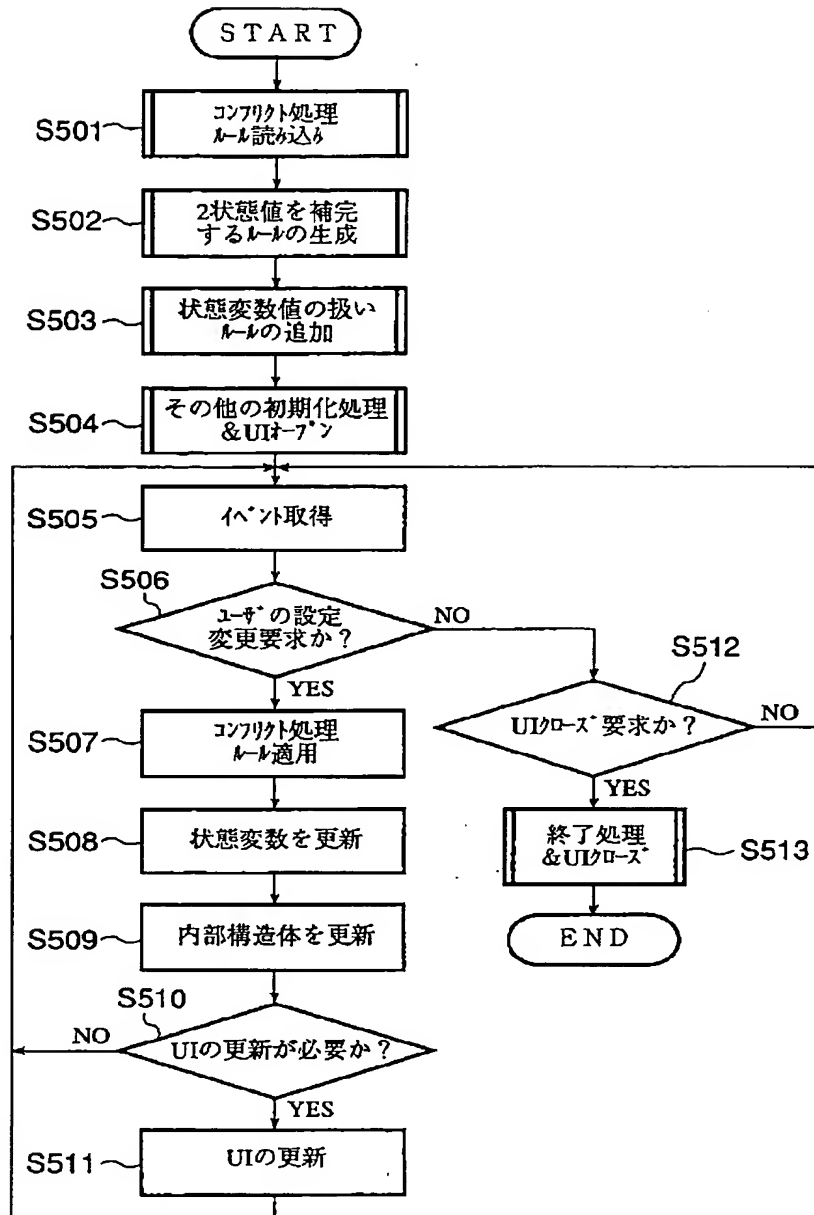
【図9】



【図10】

Collate(OFF) <- Group(ON). (1)  
 Collate(OFF) <- Staple(ON). (2)  
 Group(OFF) <- Layout(BOOKLET). (3)  
 (disable)  
 Collate(ON) <- true. (4)  
 Group(ON) <- true. (5)  
 Collate(\_X) <- status(Collate,\_X). (6)  
 Group(\_X) <- status(Group,\_X). (7)  
 Layout(\_X) <- status(Layout,\_X). (8)  
 Staple(\_X) <- status(Staple,\_X). (9)

【図5】



【図11】

Collate(OFF) <- Group(ON). (1)  
 Collate(OFF) <- Staple(ON). (2)  
 Group(OFF) <- Layout(BOOKLET). (3)  
     {disable}  
     {Message(MSG001)}  
 Collate(ON) <- true. (4)  
 Group(ON) <- true. (5)  
 Collate\_X <- status(Collate\_X). (6)  
 Group\_X <- status(Group\_X). (7)  
 Layout\_X <- status(Layout\_X). (8)  
 Staple\_X <- status(Staple\_X). (9)

【図12】

